BNL - 67832

USING A COMMERCIAL MATHEMATICS SOFTWARE PACKAGE FOR ON-LINE ANALYSIS AT THE BNL ACCELERATOR TEST FACILITY

R. Malone and X.J. Wang
National Synchrotron Light Source
Brookhaven National Laboratory
Upton, New York 11973-5000, USA

June 1999

National Synchrotron Light Source

Brookhaven National Laboratory
Operated by
Brookhaven Science Associates
Upton, NY 11973

Under Contract with the United States Department of Energy
Contract Number DE-AC02-98CH10886

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or any third party's use or the results of such use of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessary constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof or its contractors or subcontractors. The views and opinions of authors expresses herein do not necessarily state to reflect those of the United States Government or any agency thereof.

Using a Commercial Mathematics Software Package for On-line Analysis at the BNL Accelerator Test Facility

R. Malone and X.J. Wang

Brookhaven National Laboratory[1]
Upton, NY 11973-5000

Abstract

By writing both a custom Windows NT™ dynamic link library and generic companion server software, the intrinsic functions of MathSoft's Mathcad™ have been extended with new capabilities which permit direct access to the control system databases of Brookhaven National Laboratory's Accelerator Test Facility. Under this scheme, a Mathcad worksheet executing on a personal computer becomes a client which can both import and export data to a control system server via a network stream socket connection. The result is an alternative, mathematically oriented view of controlling the accelerator interactively.

I. INTRODUCTION

Over the past several years, a number of high quality, commercial software packages for mathematical and statistical analysis have become available for use on personal computers. One such package that has gained considerable popularity is Mathcad, a product of MathSoft, Inc. [1] Mathcad has distinguished itself because of its relatively low cost, extensive function library and ease of use. Its most notable feature is its free form, worksheet-like user interface where expressions are entered in two-dimensional, whiteboard-style using standard mathematical notation and operators. This is in contrast to say, spreadsheet programs where formulas are entered in a flat command-line fashion.

MathSoft markets several versions of Mathcad: standard, professional and professional academic. Only the professional versions permit users to extend the base package by adding their own customized functions. Although primarily designed for the definition of new mathematical functions, we have demonstrated that the same expansion capability can be used to provide network access to remote databases. The result is an enhanced Mathcad worksheet that can import remote data, use it as part of calculations and export the result. This technique has been used successfully to develop new on-line applications at Brookhaven National Laboratory (BNL)'s laser/linac research project, the Accelerator Test Facility (ATF). Such applications are especially attractive to scientists and engineers since accelerator algorithms can be expressed in high level notation. Modifications can be tested almost immediately, as the interpreted worksheet requires no compilation or linking.

II. OVERVIEW OF ATF CONTROL SYSTEM

The ATF computer control system is built around a VAX 4200 computer. Data acquisition is via CAMAC hardware communicating over a 5 MHz byte-serial highway. All operator displays and the main accelerator databases have been built using Vsystem™ [2]. The system features a point and click, graphical user interface with more than 800 window displays through which operators access, control and monitor the accelerator. Historically, these operator stations were X-terminals connected by Ethernet but have been replaced by personal computers running with Microsoft's Windows NT Workstation Version 4.0 [3] and X-terminal emulation. The replacement PCs have superior graphical performance over the X-terminals with the added benefit of local execution of application programs, such as Mathcad. Operators can use both the main control system displays together with Mathcad application windows without conflict.

III. IMPLEMENTATION

The Mathcad interconnection software requires two parts: a client portion for Mathcad on a local PC and a server application that resides on the main control system. Together, they carry out a classic synchronous client/server dialog over a Berkeley-style, network stream socket connection as shown in Table 1.

The client side is implemented as a Windows NT dynamic link library (DLL). MathSoft supports creation of this DLL using C/C++ compilers from Borland, Microsoft, Symantec or Watcom. The DLL in use at ATF was created using Microsoft Visual C++, Version 6.

Instructions for creating a user DLL are in the "\Doc\Mathcad Users Guide" folder of the Mathcad distribution CD-ROM. MathSoft's examples illustrate the creation of regular Windows DLLs using C. If C++ is to be used instead, we recommended creating a DLL that includes Microsoft Foundation Classes (MFC), a so-called regular MFC DLL. Such a library provides a rich set of classes from which user-created objects can enhance the Mathcad interface even further by using, for example, message boxes, file dialogs, etc.

[1] Supported by the U.S. Department of Energy under contract number DE-AC02-98CH10886

Table 1

Data Flow Between Mathcad Worksheet and Main Control System

PC EXECUTING MATHCAD			MAIN CONTROL SYSTEM	
Mathcad Worksheet	User DLL	*Network Communication*	Mathcad Server	Accelerator Database
			• Create network socket • Bind to known address • Place socket in passive mode, listening for requests	
• First instance of Mathcad begins execution • Mathcad loads user DLL(s) →	• Initialize Windows sockets • Obtain DLL instance handle • Create user error message table ←			
• Calculate worksheet • *atf_host_connect( )* called →	• Create Windows socket • Connect to remote host → • Return connection status to Mathcad ←	→ → → ← ← ←	• Accept connection request • Send acknowledgement message	
• Mathcad continues to execute worksheet				
• atf_put_* ( ) or atf_get_* ( ) called →	• Send database request message →	→ → →	• Analyze database request • Perform desired operation →	
	• Update worksheet variable(s)	← ← ←	• Send acknowledgement message ←	• Database read/write • Return status ←
• Mathcad continues to execute worksheet				
• *atf_host_disconnect* called →	• Shut down and close socket →	→ → →	• Close link and prepare for new connection	

As MathSoft provides no guidance on how to build an MFC DLL, users wishing to implement their own libraries may find remarks based on ATF experience useful:

- Use the Microsoft Application Wizard to create a project for a statically linked regular MFC DLL.
- Include the Mathcad header file *mcadincl.h* as part of the precompiled header file *stdafx.h*
- Add code to the *InitInstance* member of the application:
  - Initialize Windows sockets by calling *AfxSocketInit( )*
  - Obtain the DLL instance handle by calling *AfxGetInstanceHandle( )*
  - Define the Mathcad error message table and create user functions as in the Mathcad documentation.
- Override the *ExitInstance( )* member of the application to shutdown and close any socket created during worksheet execution.

- Create any network sockets as globals since unloading of functions can cause the socket to go out of scope resulting in unexpected disconnection from the network.
- Protect user functions from unexpected MFC assertions by adding the macro call *AFX_MANAGE_STATE(AfxGetStaticModuleState())* at the beginning of every user function.
- Add the path and filename for the user DLL to the project settings, under the Debug / Other DLLs section. This will enable use of the debugger concurrently with Mathcad.
- Use Microsoft's CSocket class for network sockets as it provides a convenient encapsulation of regular Windows sockets.

Reference [4] provides more details on many aspects of dynamic link libraries.

Once placed in the appropriate folder, Mathcad automatically will load the DLL as part of its own initialization. User functions may then be called from within a Mathcad worksheet.

Construction of the control system server is essentially standard network socket programming. See [5] for detailed explanations and code samples.

Although remote procedure calls (RPCs) would work equally well, our implementation uses sockets since the server-side code can be reused to support interoperability with other applications, notably National Instruments' LabVIEW™ [6] which has built-in support for sockets.

IV. Sample Application

At present, the ATF DLL contains some 25 functions for access to the accelerator databases. These include operations for network connections, reading and writing data of various types including real, integer, string, binary, etc., time delay and messages boxes. Figure 1 shows a Mathcad worksheet that calculates the emittance of ATF's photoelectron beam. The procedure is carried out by varying the current in a particular quadrupole magnet and measuring the size of the resulting beam spot from a digitized video frame. The Data Acquisition section of the worksheet contains function calls (prefixed by "atf_") that access the remote control system databases.

When executed,

atf_host_connect
(str2vec ("bnlatc.atf.bnl.gov"), 305, passwd)

attempts creation of a network connection to the control system on port 305 and submits a password for access. The *str2vec( )* forces Mathcad to convert the host name string to a vector of ASCII values. This is needed since all user functions must pass (and return) only numeric scalars or arrays. This is not a particular hardship since the underlying DLL functions can cast appropriately. A return value of zero from the *atf_host_connect( )* call indicates a connection has been established successfully. Should the attempt fail, the command would be highlighted in red, an error message displayed and further calculation of the worksheet suspended.

The next two statements

ptr_mps := atf_get_chix
(str2vec("DARL29;CDS;SET_CURRENT_SETPT"))

and

ptr_sigma := atf_get_chix
(str2vec("FRAME_DB::FGR1;RSX;SIGMA_X"))

retrieve database channel indices (pointers) for the quadrupole magnet power supply and the beam spot size as measured by the video frame grabber, respectively. These pointers are used in all subsequent function calls that access the database. The strings in quotation marks are the database channel names which have the general form *database_name::channel_name*. If omitted, the database name defaults to "RT_DATABASE".

The worksheet then loops over the range of quadrupole current setpoints, repeatedly executing the three statements

status ¬ atf_put_real (ptr_mps, setpt)

status ¬ atf_sleep (1000.)

sigma ¬ atf_get_real(ptr_sigma)
** pixel_cal_horiz /1000*

which write a new magnet power supply setpoint to the database, suspend worksheet execution for one second, and finally, read the spot size measured by the frame grabber, scaling it from pixels to millimeters. The back arrow is Mathcad's assignment operation inside program blocks.

Termination of the network connection occurs when calling *atf_host_disconnect (0)* =, the zero argument and equals sign being needed to force function evaluation.

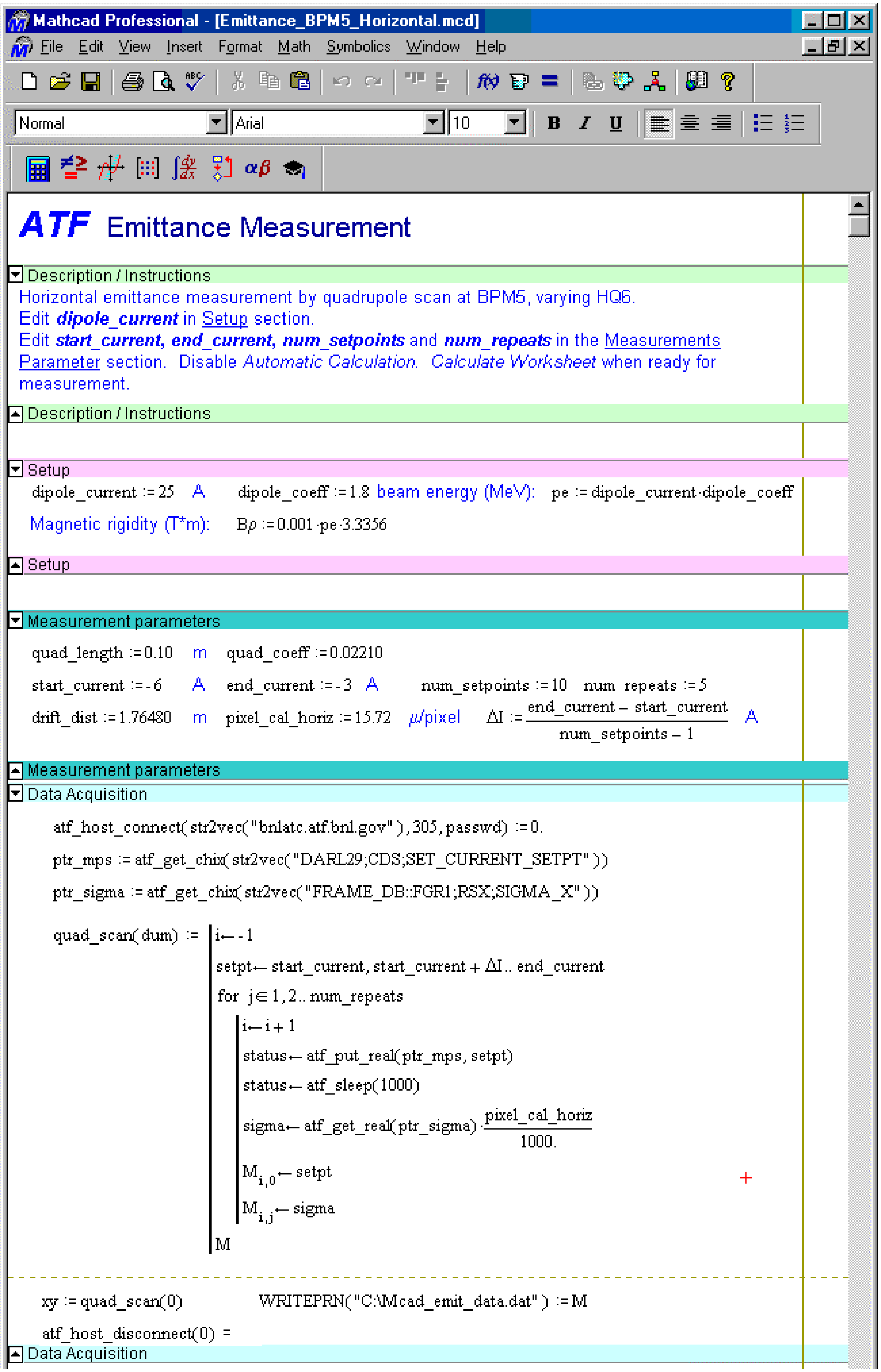
Note that the Calculation section of the worksheet has been programmed using two-dimensional, standard mathematical notations.

V. Performance / Real-time Issues

The non-determinism of Ethernet timing combined with Mathcad's interpreted (vs. compiled) worksheet preclude use of this technique for hard real time problems. Using a 400 MHz Pentium II PC with Windows NT Workstation communicating to the main VAX over a 10 MHz Ethernet, we have measured the time needed to write a single real value to the database at ~35 msec. This includes the time to write the value, update the remote database and receive a return acknowledgment message. This figure probably could be improved, but we have made no particular attempt to do so since this throughput is adequate for many of ATF's needs.

As ATF is a pulsed machine, the more serious concern is that a set of measurements imported into a worksheet did indeed come from the same beam pulse. To address this issue, a set of list handling functions is defined. The general call sequence is:

```
atf_list_define ( list_number )
atf_list_add ( pointer_1 )
atf_list_add ( pointer_2 )

atf_list_add ( pointer_n )
atf_list_end ( list_number )
atf_list_send ( list_number )
```


Mathcad Professional - [Emittance_BPM5_Horizontal.mcd]
File Edit View Insert Format Math Symbolics Window Help
Normal
Arial
10
B I U

ATF Emittance Measurement

Description / Instructions

Horizontal emittance measurement by quadrupole scan at BPM5, varying HQ6.
Edit ***dipole_current*** in Setup section.
Edit ***start_current, end_current, num_setpoints*** and ***num_repeats*** in the Measurements Parameter section. Disable *Automatic Calculation.* *Calculate Worksheet* when ready for measurement.

Description / Instructions

Setup

dipole_current := 25 A dipole_coeff := 1.8 beam energy (MeV): pe := dipole_current·dipole_coeff

Magnetic rigidity (T*m): $B\rho := 0.001 \cdot pe \cdot 3.3356$

Setup

Measurement parameters

quad_length := 0.10 m quad_coeff := 0.02210

start_current := -6 A end_current := -3 A num_setpoints := 10 num_repeats := 5

drift_dist := 1.76480 m pixel_cal_horiz := 15.72 μ/pixel $\Delta I := \frac{\text{end\_current} - \text{start\_current}}{\text{num\_setpoints} - 1}$ A

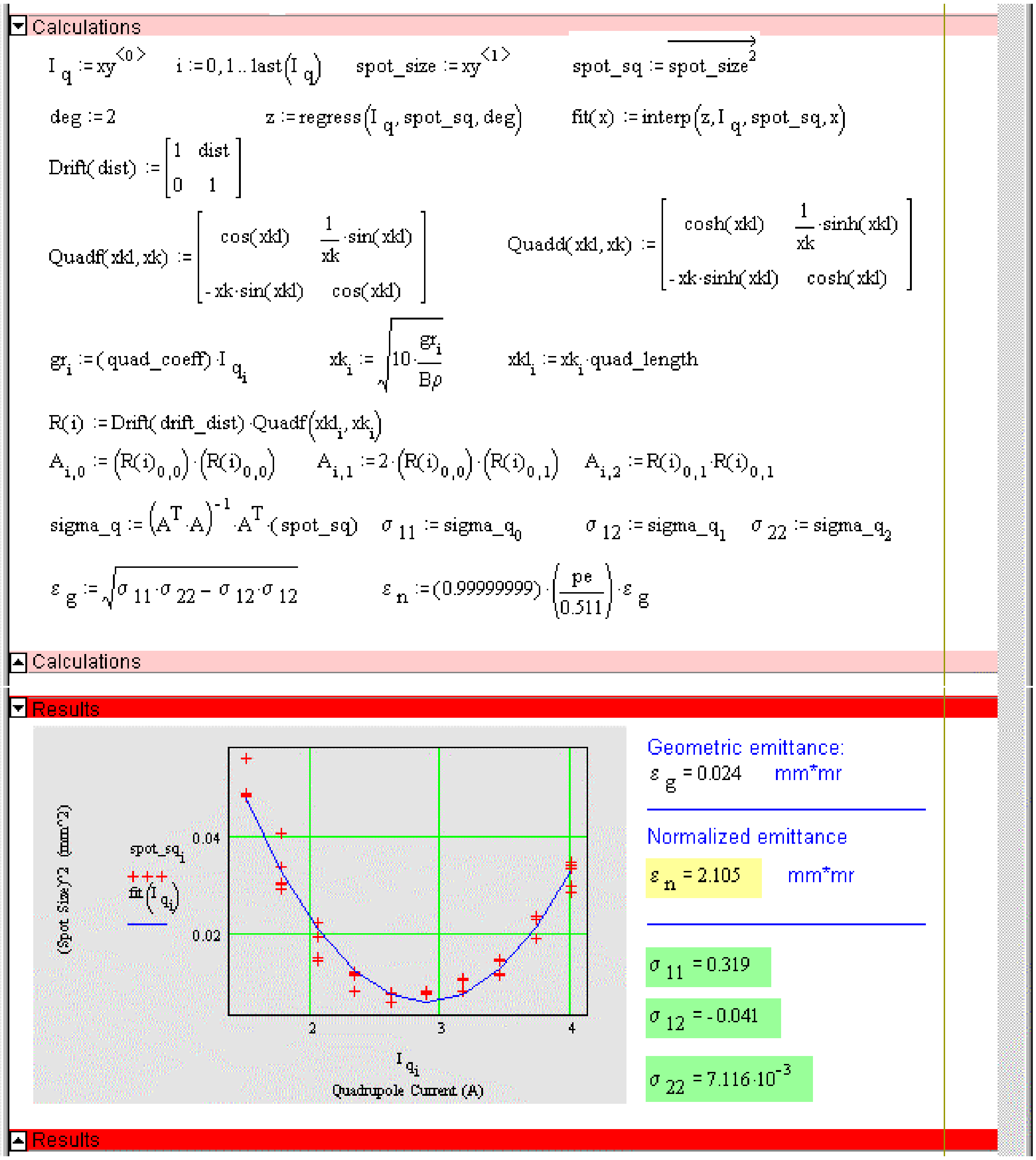
Measurement parameters

Data Acquisition

```
atf_host_connect( str2vec( "bnlatc.atf.bnl.gov" ), 305, passwd) := 0.

ptr_mps := atf_get_chix( str2vec( "DARL29;CDS;SET_CURRENT_SETPT" ))

ptr_sigma := atf_get_chix( str2vec( "FRAME_DB::FGR1;RSX;SIGMA_X" ))

quad_scan( dum) := | i ← -1
                   | setpt ← start_current, start_current + ΔI .. end_current
                   | for j ∈ 1, 2 .. num_repeats
                   |     | i ← i + 1
                   |     | status ← atf_put_real( ptr_mps, setpt)
                   |     | status ← atf_sleep( 1000)
                   |     | sigma ← atf_get_real( ptr_sigma) · $\frac{\text{pixel\_cal\_horiz}}{1000.}$
                   |     | $M_{i,0}$ ← setpt
                   |     | $M_{i,j}$ ← sigma
                   | M

xy := quad_scan( 0)            WRITEPRN( "C:\Mcad_emit_data.dat" ) := M

atf_host_disconnect( 0) =
```

Data Acquisition

Figure 1: Sample Application - Emittance Measurement



Calculations

$I_q := xy^{\langle 0 \rangle}$ $\quad i := 0, 1 .. \mathrm{last}(I_q)$ $\quad spot\_size := xy^{\langle 1 \rangle}$ $\quad spot\_sq := \overrightarrow{spot\_size^2}$

$deg := 2$ $\quad z := \mathrm{regress}(I_q, spot\_sq, deg)$ $\quad \mathrm{fit}(x) := \mathrm{interp}(z, I_q, spot\_sq, x)$

$$\mathrm{Drift}(dist) := \begin{bmatrix} 1 & dist \\ 0 & 1 \end{bmatrix}$$

$$\mathrm{Quadf}(xkl, xk) := \begin{bmatrix} \cos(xkl) & \frac{1}{xk} \cdot \sin(xkl) \\ -xk \cdot \sin(xkl) & \cos(xkl) \end{bmatrix} \qquad \mathrm{Quadd}(xkl, xk) := \begin{bmatrix} \cosh(xkl) & \frac{1}{xk} \cdot \sinh(xkl) \\ -xk \cdot \sinh(xkl) & \cosh(xkl) \end{bmatrix}$$

$$gr_i := (quad\_coeff) \cdot I_{q_i} \qquad xk_i := \sqrt{10 \cdot \frac{gr_i}{B\rho}} \qquad xkl_i := xk_i \cdot quad\_length$$

$$R(i) := \mathrm{Drift}(drift\_dist) \cdot \mathrm{Quadf}(xkl_i, xk_i)$$

$$A_{i,0} := \left(R(i)_{0,0}\right) \cdot \left(R(i)_{0,0}\right) \qquad A_{i,1} := 2 \cdot \left(R(i)_{0,0}\right) \cdot \left(R(i)_{0,1}\right) \qquad A_{i,2} := R(i)_{0,1} \cdot R(i)_{0,1}$$

$$sigma\_q := \left(A^T \cdot A\right)^{-1} \cdot A^T \cdot (spot\_sq) \qquad \sigma_{11} := sigma\_q_0 \qquad \sigma_{12} := sigma\_q_1 \qquad \sigma_{22} := sigma\_q_2$$

$$\varepsilon_g := \sqrt{\sigma_{11} \cdot \sigma_{22} - \sigma_{12} \cdot \sigma_{12}} \qquad \varepsilon_n := (0.99999999) \cdot \left(\frac{pe}{0.511}\right) \cdot \varepsilon_g$$

Calculations

Results

Geometric emittance:

$\varepsilon_g = 0.024$ mm*mr

Normalized emittance

$\varepsilon_n = 2.105$ mm*mr

$\sigma_{11} = 0.319$

$\sigma_{12} = -0.041$

$\sigma_{22} = 7.116 \cdot 10^{-3}$

Results

Figure 1 (continued): Sample Application – Emittance Measurement

Atf_list_send(list_number) send a list of database pointers to the remote host which keeps a copy, indexed by the list number. When worksheet calculations require measurements from a single beam pulse, a call to *atf_list_execute (list_number)* is made to signal the control system to collect the desired items during one beam cycle and to return them to Mathcad as a real vector. As the main host is fully aware of the facility timing cycles, it can guarantee all requested items have come from the same beam pulse.

VI. Conclusions

We have successfully integrated Mathcad into the on-line analysis tools in use at the Accelerator Test Facility. The technique we have described need not be limited to network access of remote databases. Similar dynamic link libraries could be constructed, for example, to read and analyze the data from local instrumentation connected directly to a personal computer by IEEE-488 or RS-232 interfaces.

VII. Acknowledgement

We thank our colleague, V. Yakimenko, who enthusiastically tested some of the early versions of our library.

VIII. References

[1] MathSoft, Inc., 101 Main Street, Cambridge, MA 02142-1521

[2] Vista Control Systems, Inc., 176 Central Park Square, Los Alamos, NM 87554

[3] Microsoft, Inc., One Microsoft Way, Redmond, WA 98052-6399

[4] M. Blaszczak, Professional MFC with Visual C++ 5, Chicago: Wrox Press, Inc., 1997, pp. 579-614.

[5] D.E. Comer and D.L. Stevens, Internetworking With TCP/IP Volume III: Client-Server Programming and Applications, BSD Socket Version, Englewood Cliffs: Prentice Hall, 1993, pp. 75-91.

[6] National Instruments Corporation, 11500 North Mopac Expressway, Austin TX 78759-3504